Colorado School of Mines
Department of Mechanical Engineering
MEGN 540: Mechatronics

Laboratory 0: Getting Started   Version 1.0

Objective: Become familiar with the lab setup and software tools and control the LED on the Zumo board to print out 'Hello World!' in Morse code.

Equipment:

· Lab Kit   Mouse   Keyboard   Monitor   Internet Connection

Background:

The Pololu Zumo car is based on the Lenardo Arduino development boards and provides a great opportunity to practice embedded C programming on an interesting platform. This board has many functionalities including an 8-bit Atmel microcontroller, I/O ports, USB connectivity, and a DC power input. Arduino has also released an associated development environment (with programs called "sketches"), which comes bundled with pre-written functions for common microcontroller programming routines. These are available for the Pololu, but abstract away much of the critical learning for embedded development. However, we will be bypassing the Arduino development environment and we will be programming the microcontroller directly using embedded C and the avrdude programming tool.

The Kit

The core of the our mechatronics kit is the Zumo Pololu robotic car. Specifically, we'll be using the Zumo 32U4 Robot with 75:1 motors (https://www.pololu.com/product/3126). This platform comes with many sensors and interfaces, only some of which we'll use in this lab. In particular it incorporates an atmega32U4 microprocessor, which we'll be coding, and the A-Start boot loader (https://github.com/pololu/a-star) that allows programming over the USB interface.

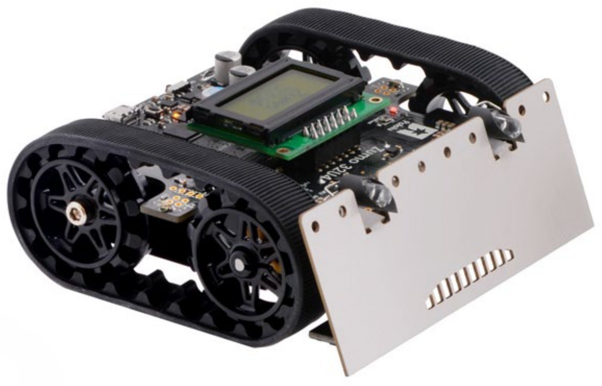

Figure 1. The Zumo Pololu

Attached to the board is a Raspberry PI 4+ (https://www.raspberrypi.org/products/raspberry-pi-4-model-b/) running the Raspbian linux based operating system (https://www.raspbian.org/). This will act as a control interface for programming and interactions, as well as enable hi-level user inputs for a fully robotic style system.

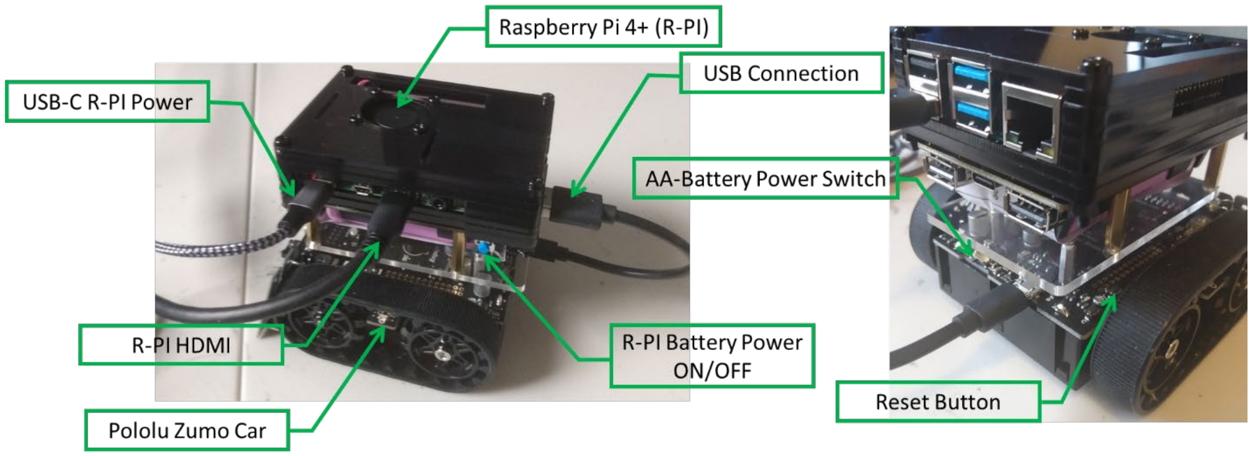

Figure 2. Platform

All code and labs will be available from Github: https://github.com/M3R-CSM/MEGN540. The version control tool git will be used to update lab template codes and setup files. As a student, you'll need to develop a passing familiarity with git, for reference see https://education.github.com/git-cheat-sheet-education.pdf.

Environment Setup

The kit has come pre-loaded with much of the software necessary. Though, there may have been some changes since it was initially setup.

1. Plug in the R-PI to USB-C power using the provided cables, attach to a monitor, mouse, and keyboard. Connect it to the internet.
2. Once ready booted, open a terminal.
   ```
   >>	ctrl-alt-t
   ```
3. Navigate to the Mechatronics folder
   ```
   >>	cd Desktop/MEGN540
   ```
4. Update the code to the current version on the github repository.
   ```
   >>	git pull origin main
   ```
5. Run the setup script (if prompted for a password use MEGN540!)
   ```
   >>	sh MEGN540_setup.sh
   ```
6. Attach the short USB from the R-PI to the Zumo Car
7. Put 4 (charged) AA batters into the bottom of the Zumo Car.
8. Turn 'On' the Zumo Car
9. In the terminal type:
   ```
   >> ls /dev/ttyZ*
   ```
   You should see either a /dev/ttyZumoCar or /dev/ttyZumoCarAVR output

---

Uploading your program using AVRDUDE

1. In the terminal, navigate to the Lab0-Blink folder [cd ~/Desktop/MEGN540/Lab0-Blink]. Everything is configured using a Makefile, take a moment to open it in an editor and note where the lines for adding your code exist (below the line that says: # List C source files here. (C dependencies are automatically generated.)). This is where you'd add any special c files you write for the labs. Each lab will have an associated Makefile with the template code already setup to help you with the programming.
2. Make the project
   >> make
3. Program the Board
   a. Type (but don't hit enter yet)
      make program
   b. Press the Reset button on the Zumo car twice within 750ms. If successful, a yellow LED light will pulse telling you it's ready to program. This sometimes may take more than one try…
   c. Press enter on your 'make program' command. The system will use AVRDUDE to program the device and the output from the command should indicate all things are well.
   d. If successful the yellow LED should start blinking 'SOS' in Morse Code. Time to help your robot!

Assignment:

The main assignment in this lab is to control the blinking of the LED using a non-event driven programming approach. In a future lab, you'll revisit this to make it compatible with an event-driven approach. For this lab, you'll be using the code provided in the led_interface.h/c and will be modifying the LAB0-blink.c file, which contains your main function.

1. Review the provided functions, understand what they are doing, and how their called.
   a. This includes pint setups and register calls for the LED. Look on the pin-out diagram for the zumo car and see if you can relate the LED settings to the car's wiring schematic. (https://www.pololu.com/file/0J862/zumo-32u4-schematic-diagram.pdf)
2. Create a function in LAB0-blink.c that takes a c-string (char* str) (https://en.wikibooks.org/wiki/C_Programming/String_manipulation) and then causes the LED to blink the Morse code pattern for the string. Remember, c-strings are always one-byte longer than the characters and are terminated with a NULL (0) byte, so you can iterate with

```
while( str[i]!=0 ){
        DO SOMETHING WITH str[i]
        i++;
}
```

3. Recorded a Video of your robot writing blinking out the message: Hello World! With a 100ms Morse code ‘dot’ length.

Deliverables:

1. Demo: Make a 1-2-Minute Video that talks through your code and how you completed the lab. (free voice and screen capture video software: https://obsproject.com/)
2. Demo: Upload a video of your car blinking out the message.
3. Lab Report: Prepare a summary of what you have completed for this assignment. Follow a typical lab report format. Make sure that you:
    a. Include the information that you found important and critical while completing this lab, and would be useful if you wanted to replicate it in the future.
    b. Indicate which pins on the electrical schematic were used in this project, and how you can tell which registers you needed to initialize and set.
    c. Discuss what you’d need functionally to get this to work in an event-driven (e.g., non-blocking) manner.

Tutorials and Other Resources:

There are many tutorials and web sites that focus on microcontroller programming. The following are ones that were found (by the instructor and students who took this class) to be particularly helpful and to follow a clear presentation. You may want to go through these tutorials before you start working on the assignment. Additional references at the end are not required and you can check them only if you are interested.

As you work on the lab assignments, if you come across additional resources/sites which you find to be particularly helpful, email them to the instructor, so they can be shared with the others in the class in current or future semesters, and start a Discussion thread to let your fellow classmates know!

An Overview of Microcontrollers and Microcontroller Programming

The following provides a brief introduction to microcontrollers, steps involved in developing code for a microcontroller, embedding/writing it onto the microcontroller, and AVR microcontrollers and their peripherals. Some of this content will be covered in class.

"Basics of Microcontrollers"
http://maxembedded.com/2011/06/basics-of-microcontrollers/

" AVR Basics"
http://maxembedded.com/2011/06/avr-basics/

" Port operations in AVR"

http://maxembedded.com/2011/06/port-operations-in-avr/